Weighted Model Counting Without Parameter Variables

17th February 2021

1 Introduction

Notation. For any propositional formula ϕ and $p, q \in \mathbb{R}$, let $[\phi]^p_q \colon 2^X \to \mathbb{R}$ be a pseudo-Boolean function defined as

$$[\phi]^p_q(Y) \coloneqq \begin{cases} p & \text{if } Y \models \phi \\ q & \text{otherwise} \end{cases}$$

for any $Y \subseteq X$.

Definition 1 (old WMC instance)**.** An *old WMC instance* is a tuple (ϕ, X_I, X_P, w), where X_I is the set of indicator variables, X_P is the set of parameter variables (with $X_I \cap X_P = \emptyset$), ϕ is a propositional formula in CNF over $X_I \cup X_P$, and $w \colon X_I \cup X_P \cup \{\neg x \mid x \in X_I \cup X_P\} \to \mathbb{R}$ is the weight function. The *answer* of the instance is $\sum_{Y \models \phi} \prod_{Y \models l} w(l)$.

Remark. Encodings such as `cd05` and `cd06` are *not* WMC encodings. Instead, they encode Bayesian network inference into instances of the *minimum-cardinality* WMC problem, where the answer is defined to be $\sum_{Y \models \phi,\ |Y| = k} \prod_{Y \models l} w(l)$, where $k = \min_{Y \models \phi,\ Y \neq \emptyset} |Y|$, if k exists, otherwise the answer is zero. This additional condition on model cardinality becomes necessary because these encodings eliminate clauses of the form $p \Rightarrow i$, where $p \in X_P$ is a parameter variable, and $i \in X_I$ is an indicator variable. Nonetheless, our transformation algorithm still works on such encodings, although the experimental results are discouraging because they use approximately twice as many indicator variables. For instance, each binary variable of a Bayesian network is encoded using two indicator variables while one would suffice.

Definition 2 (new WMC instance)**.** A *new WMC instance* is a tuple (F, X, ω), where X is the set of variables, F is a set of pseudo-Boolean functions $2^X \to \mathbb{R}$, and $\omega \in \mathbb{R}$ is the scaling factor. The *answer* of the instance is $\omega \cdot \left(\exists_X \prod_{f \in F} f\right)(\emptyset)$.

References.

- Related work without publicly available implementations:
  - direct compilation to SDDs [2]
  - direct compilation to PSDDs, also eliminating parameter variables (a thesis)
  - maybe two more papers

Notes.

- Apparently, the DPMC paper already shows that taking the first offered decomposition tree is best.
- It is already well-known that WMC is FPT.

2 Parameter Variable Elimination

Notes.

- Let X_P be the set of parameter variable and X_I be the set of indicator variables.
- Parameter variables are either taken from the LMAP file (for encodings produced by Ace) or assumed to be the variables that have both weights equal to 1.
- If a parameter variable in a clause is 'negated', we can ignore the clause. We assume that there are no clauses with more than one instance of parameter variables.
- The second **foreach** loop can be performed in constant time by representing ϕ' as a list and assuming that the two 'clauses' are adjacent in that list (and incorporating it into the first loop).
- The d map is constructed in $\mathcal{O}(|X_P| \log |X_P|)$ time (we want to use a data structure based on binary search trees rather than hashing).
- `rename` can be implemented in $\mathcal{O}(\log |X_P|)$ time.
- This may look like preprocessing, but all the transformations are local and thus can be incorporated into an encoding algorithm with no slowdown. In fact, if anything, the resulting algorithm would be slightly faster, as it would have less data to output.

Algorithm 1: WMC instance transformation

Data: an (old-format) WMC instance (ϕ, X_I, X_P, w)
Result: a (new-format) WMC instance (F, X, ω)
$F \leftarrow \emptyset$;
$\omega \leftarrow 1$;
let $d \colon X_P \to \mathbb{N}$ be defined as $p \mapsto |\{o \in X_P \mid o \leq p\}|$;
foreach *clause* $c \in \phi$ **do**
    if $c \cap X_P = \{p\}$ *for some* p **and** $w(p) \neq 1$ **then**
        if $|c| = 1$ **then**
            $\omega \leftarrow \omega \times w(p)$;
        else
            $F \leftarrow F \cup \left\{ \left[\bigwedge_{l \in c \setminus \{p\}} \neg l \right]_1^{w(p)} \right\}$;
    else if $\{p \mid \neg p \in c\} \cap X_P = \emptyset$ **then**
        $F \leftarrow F \cup \{[c]_0^1\}$;
foreach *indicator variable* $v \in X_I$ **do**
    if $\{[v]_1^p, [\neg v]_1^q\} \subseteq F$ *for some* p *and* q **then**
        $F \leftarrow F \setminus \{[v]_1^p, [\neg v]_1^q\} \cup \{[v]_q^p\}$;
replace every variable v in F with `rename(`v`)`;
return (F, X_I, ω);
Function `rename(`v`)`**:**
    $S \leftarrow \{u \in X_P \mid u \leq v\}$;
    if $S = \emptyset$ **then return** v;
    return $v - d(\max S)$;

2.1 Correctness

Theorem 1 (Early Projection [3, 4], verbatim)**.** *Let X and Y be sets of variables. For all functions $f\colon 2^X \to \mathbb{R}$ and $g\colon 2^Y \to \mathbb{R}$, if $x \in X \setminus Y$, then $\exists_x(f \cdot g) = (\exists_x f) \cdot g$.*

Lemma 1. *For any pseudo-Boolean function $f\colon 2^X \to \mathbb{R}$, $(\exists_X f)(\emptyset) = \sum_{Y \subseteq X} f(Y)$.*

Proof. For base case, let $X = \{x\}$. Then

$$\exists_x f = f|_{x=1} + f|_{x=0},$$

by Definition 8 and so

$$(\exists_x f)(\emptyset) = f|_{x=1}(\emptyset) + f|_{x=0}(\emptyset)$$

by Definition 6. Then

$$f|_{x=1}(\emptyset) = f(\{x\}), \quad \text{and} \quad f|_{x=0}(\emptyset) = f(\emptyset)$$

by Definition 7. It follows that

$$(\exists_x f)(\emptyset) = f(\{x\}) + f(\emptyset) = \sum_{Y \subseteq \{x\}} f(Y)$$

as required. This easily extends to $|X| > 1$ by the definition of projection on sets of variables. □

Proposition 1. *Let (ϕ, X_I, X_P, w) be an old WMC instance. Then*

$$\left(\left\{[c]_0^1 \mid c \in \phi\right\} \cup \left\{[x]_{w(\neg x)}^{w(x)} \,\middle|\, x \in X_I \cup X_P\right\}, X_I \cup X_P, 1\right) \tag{1}$$

is a new WMC instance with the same answer.

Proof. The answer of Eq. (1) is

$$\begin{aligned}\left(\exists_{X_I \cup X_P} \prod_{c \in \phi} [c]_0^1 \prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)}\right)(\emptyset) &= \sum_{Y \subseteq X_I \cup X_P} \left(\prod_{c \in \phi} [c]_0^1 \prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)}\right)(Y) \\ &= \sum_{Y \subseteq X_I \cup X_P} \underbrace{\left(\prod_{c \in \phi} [c]_0^1\right)}_{f}(Y) \underbrace{\left(\prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)}\right)}_{g}(Y)\end{aligned}$$

by Definitions 2 and 6 and Lemma 1. Note that

$$f(Y) = \begin{cases} 1 & \text{if } Y \models \phi, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad g(Y) = \prod_{Y \models l} w(l),$$

which means that

$$\sum_{Y \subseteq X_I \cup X_P} f(Y) g(Y) = \sum_{Y \models \phi} \prod_{Y \models l} w(l)$$

as required. □

Theorem 2 (Correctness)**.** *Algorithm 1, when given an old WMC instance (ϕ, X_I, X_P, w), returns a new WMC instance with the same answer, provided the following conditions are satisfied:*

1. *for all indicator variables $i \in X_I$, $w(i) = w(\neg i) = 1$,*
2. *and either*

(a) *for all parameter variables $p \in X_P$, there is a family of literals $(l_i)_{i=1}^n$ such that*

i. $w(\neg p) = 1$,

ii. $l_i \in X_I$ *or* $\neg l_i \in X_I$ *for all* $i = 1, \dots, n$,

iii. *and* $\{c \in \phi \mid p \in c \text{ or } \neg p \in c\} = \{p \lor \bigvee_{i=1}^n \neg l_i\} \cup \{l_i \lor \neg p \mid i = 1, \dots, n\}$;

(b) *or*

i. $w(p) + w(\neg p) = 1$,

ii. *there is no clause $c \in \phi$ with $\neg p \in c$,*

iii. *and for any pair of clauses $c, d \in \phi$ such that $c \neq d$, $p \in c$ and $p \in d$,* $\texttt{Vars}(c) \cap \texttt{Vars}(d) = \{p\}$.

Proof. First, let us assume that Condition 2a applies and consider the collective effect of the loop on Lines 4 to 11 for: 1) clauses that have a positive parameter variable, 2) clauses that have a negative parameter variable (removed), 3) clauses that don't have parameter variables (left unchanged). □

Notes.

- The family of literals could be empty.
- This just says that p is equivalent to a conjunction.
- The first group of conditions applies to `d02`, while the second group applies to `bklm16`.
- For `cd05` and `cd06`, condition $2bi$ should be replaced with $w(\neg p) = 1$.

3 Parameterised Complexity of DPMC

Notes.

- Summary of results
  - We establish DPMC inference as fixed-parameter tractable.
  - We experimentally show that DPMC is best on low-to-moderate treewidth instances, and cd06+c2d overtakes on higher treewidth instances.
- By DPMC, we always mean DMC+lg.
- The constant is inspired by the `bklm16` encoding.

TODO

- Do I need to formally consider extending a pseudo-Boolean function to a bigger domain?
- Introduce all the notation surrounding graphs ($\mathcal{V}$, $\mathcal{E}$ $\mathcal{L}$, $\mathcal{C}$, etc.) (and be consistent with its usage) and formulas (variable, literal, clause, CNF, etc.)
- Use Theorem 5 to show the connection between primal treewidth and the width of the PJT (linking the complexity of BN inference with the DPMC complexity).
- Come up with better names for old/new WMC instances.
- Define:
  - scalar operations on ADDs,
  - an ADD as a DAG,

– What does it mean for an ADD to 'have' variables? Maybe refer to pseudo-Boolean function sensitivity.
– primal graph of a CNF formula (a.k.a. Gaifman/(variable) interaction/connectivity/clique/representing graph),
– Bayesian network (I already have a definition of these last two),
– moralisation of a Bayesian network (or of any DAG),
– projection of a set of variables.

Lemma 2. *An ADD with n variables has $\mathcal{O}(2^n)$ nodes.*

Proof. An ADD with n variables can be, at most, a complete binary tree of height $n+1$ (as measured by the number of vertices). It would then have $2^0 + 2^1 + \cdots + 2^n = 2^{n+1} = \mathcal{O}(2^n)$ nodes. □

Lemma 3. *Let ϕ be a conjunction of n literals. Then the ADD representation of $[\phi]^p_q$ can be constructed in $\mathcal{O}(2^n)$ time for any $p, q \in \mathbb{R}$ such that $p \neq q$.*

Proof. The ADD representation of ϕ itself can be constructed with a sequence of $n-1$ calls to `apply` with one of the two operands in each call always a literal. The number of variables in the other operand then follows the sequence $1, 2, 3, \dots, n-1$. By Lemma 2, the numbers of nodes in the ADD representations of these operands is then $\mathcal{O}(2^1), \mathcal{O}(2^2), \dots, \mathcal{O}(2^{n-1})$. Since one of the operands is of constant size, the overall time complexity of all calls to `apply` is then

$$\mathcal{O}(2^1) + \mathcal{O}(2^2) + \cdots + \mathcal{O}(2^{n-1}) = \mathcal{O}(2^n).$$

Let α be the ADD representation of ϕ. Then the ADD representation of $[\phi]^p_q$ is $(p-q)\alpha + q$. As scalar operations can obviously be implemented in linear time, the overall complexity remains $\mathcal{O}(2^n)$. □

Theorem 3 ([5], rephrased)**.** *BN Inference (for all algorithms that accept arbitrary instances) has a lower bound that's linear in the size of the BN and exponential in the treewidth of its moralisation (provide the exact formula).*

Definition 3 ([4], with some changes)**.** Let X be a set of Boolean variables, and F be a set of pseudo-Boolean functions $2^X \to \mathbb{R}$. A *project-join tree* (PJT) of F is a tuple (T, r, γ, π) where:

- T is a tree with root $r \in \mathcal{V}(T)$,
- $\gamma \colon \mathcal{L}(T) \to F$ is a bijection between the leaves of T and the pseudo-Boolean functions in F, and
- $\pi \colon \mathcal{V}(T) \setminus \mathcal{L}(T) \to 2^X$ is a labelling function on internal nodes.

Moreover, (T, r, γ, π) must satisfy the following two properties:

1. $\{\pi(n) : n \in \mathcal{V}(T) \setminus \mathcal{L}(T)\}$ is a partition of X, and
2. for each internal node $n \in \mathcal{V}(T) \setminus \mathcal{L}(T)$, variable $x \in \pi(n)$, and 'clause' $c \in F$ such that x appears in c, the leaf node $\gamma^{-1}(c)$ must be a descendant of n in T.

Definition 4 ([4])**.**

$$\texttt{Vars}(n) \coloneqq \begin{cases} \texttt{Vars}(\gamma(n)) & \text{if } n \in \mathcal{L}(T) \\ \left(\bigcup_{o \in \mathcal{C}(n)} \texttt{Vars}(o)\right) \setminus \pi(n) & \text{if } n \notin \mathcal{L}(T). \end{cases}$$

$$\texttt{size}(n) \coloneqq \begin{cases} |\texttt{Vars}(n)| & \text{if } n \in \mathcal{L}(T) \\ |\texttt{Vars}(n) \cup \pi(n)| & \text{if } n \notin \mathcal{L}(T). \end{cases}$$

(Note that $\texttt{size}(n)$ is the number of variables that can appear during the computation of $\delta(n)$ (excluding recursive calls).) The *width* of a PJT (T, r, γ, π) is $\texttt{width}(T) \coloneqq \max_{n \in \mathcal{V}(T)} \texttt{size}(n)$.

Theorem 4 ([4], rephrased, 'ADD width is equal to the treewidth of the primal graph')**.** *Given a CNF formula ϕ with a tree decomposition of its primal graph of width w, Algorithm 2 (from [4]) returns a PJT of ϕ of width at most $w + 1$.*

Definition 5 ([6], rephrased)**.** A *tree decomposition* of a graph G is a pair (T, χ), where T is a tree and $\chi\colon \mathcal{V}(T) \to 2^{\mathcal{V}(G)}$ is a labelling function, with the following properties:

- $\bigcup_{t\in\mathcal{V}(T)} \chi(t) = \mathcal{V}(G)$;
- for every edge $e \in \mathcal{E}(G)$, there exists $t \in \mathcal{V}(T)$ such that e has both endpoints in $\chi(t)$;
- for all $t, t', t'' \in \mathcal{V}(T)$, if t' is on the path between t and t'', then $\chi(t) \cap \chi(t'') \subseteq \chi(t')$.

The *width* of tree decomposition (T, χ) is $\max_{t\in\mathcal{V}(T)} |\chi(t)| - 1)$. The *treewidth* of graph G is the smallest w such that G has a tree decomposition of width w.

Definition 6. Let $f, g\colon 2^X \to \mathbb{R}$ be pseudo-Boolean functions. Operations such as addition and multiplication are defined pointwise as

$$(f + g)(Y) \coloneqq f(Y) + g(Y),$$

and

$$(fg)(Y) \coloneqq f(Y)g(Y)$$

for all $Y \subseteq X$.

Definition 7. Let $f\colon 2^X \to \mathbb{R}$ be a pseudo-Boolean function, and $x \in X$. Then $f|_{x=0}, f|_{x=1}\colon 2^X \to \mathbb{R}$ are *restrictions* of f defined as

$$f|_{x=0}(Y) \coloneqq f(Y \setminus \{x\}),$$

and

$$f|_{x=1}(Y) \coloneqq f(Y \cup \{x\})$$

for all $Y \subseteq X$.

Definition 8. Let X be a set. For any $x \in X$, *projection* $\exists_x$ is an endomorphism $\exists_x\colon \mathbb{R}^{2^X} \to \mathbb{R}^{2^X}$ defined as

$$\exists_x f = f|_{x=1} + f|_{x=0}$$

for any $f\colon 2^X \to \mathbb{R}$.

Lemma 4. *Let $f, g\colon 2^X \to \mathbb{R}$ be pseudo-Boolean functions represented by ADDs with n and m nodes, respectively, and $x \in X$. Then $f + g$ and fg can be computed in $\mathcal{O}(mn)$ time, and $\exists_x f$ can be computed in $\mathcal{O}(n^2)$ time.*

Proof. Addition and multiplication are implemented by `apply` algorithm which takes $\mathcal{O}(mn)$ time [1]. Projection consists of two restrictions and an addition by Definition 8. The computational complexity of restriction is dominated by the reduction operation that transforms a decision diagram into a minimal canonical form [1]. While the original reduction algorithm had a $\mathcal{O}(n \log n)$ complexity, caching can reduce it to $\mathcal{O}(n)$ [7]. Either way, the complexity of projection is still $\mathcal{O}(n^2)$. □

Definition 9. Let (T, r, γ, π) be a PJT, and X be the set of variables. The functionality of DPMC execution can be represented by $\delta(r)$, where $\delta\colon \mathcal{V}(T) \to \mathbb{R}^{2^X}$ is a recursive function defined as

$$\delta(t) = \begin{cases} \gamma(t) & \text{if } t \in \mathcal{L}(T) \\ \exists_{\pi(t)} \prod_{u\in\mathcal{C}(t)} \delta(u) & \text{otherwise.} \end{cases} \tag{2}$$

The range of $\delta(r)$ then contains a single real number, i.e., the answer.

Theorem 5 (Theorem 4 in [4], almost verbatim)**.** *Let ϕ be a propositional formula in CNF over a set X of variables and (S, χ) be a tree decomposition of the primal graph of ϕ of width w. Then Algorithm 2 returns a PJT of ϕ of width at most $w + 1$.*

Theorem 6. *Let (F, X, ω) be a WMC instance, and (T, r, γ, π) be its PJT of width k. Then* DPMC *execution is fixed-parameter tractable with respect to k. Specifically,* DPMC *execution time complexity is $\mathcal{O}(4^k nm)$, where $n = |\mathcal{L}(T)| = |F|$, and $m = |X|$ is the number of variables.*

Proof.

- We consider the complexity of constructing ADD representations of the pseudo-Boolean functions in F and the complexity of multiplication and projection operations throughout the recursive calls of δ in Definition 9.
- Let $t \in \mathcal{L}(T)$ be a leaf. Note that $|\texttt{Vars}(\gamma(t))| \leq k$ by Definition 4. Assuming that clauses have no repeating variables, it takes $\mathcal{O}(2^k)$ time to construct $\delta(t)$ by Lemma 3 and $\mathcal{O}(2^k n)$ time to construct all leaves of the PJT.
- Now let $t \in \mathcal{V}(T) \setminus \mathcal{L}(T)$ be an internal vertex, in which case we perform $|\mathcal{C}(t)| - 1$ multiplications and $|\pi(t)|$ projections. Note that $|\mathcal{C}(t)| \leq |\mathcal{L}(T)| = n$, $|\pi(t)| \leq k$, and the ADDs involved can have up to k variables. This means that each ADD has $\mathcal{O}(2^k)$ nodes (regardless of whether it comes from $\mathcal{C}(t)$ or from multiplications). Each such multiplication then takes $\mathcal{O}(4^k)$ time by Lemma 4, and so all multiplications will take $\mathcal{O}(4^k n)$ time for t and $\mathcal{O}(4^k nm)$ time across all $t \in \mathcal{V}(T) \setminus \mathcal{L}(T)$ since the number of vertices in the PJT is bounded by the number of variables.
- Each variable is projected exactly once and is always projected from an ADD with at most $\mathcal{O}(2^k)$ nodes. Thus, the time complexity of projecting all variables is $\mathcal{O}(4^k m)$.
- In total, we get $\mathcal{O}(2^k n)$ time for constructing initial ADDs, $\mathcal{O}(4^k nm)$ for multiplications, and $\mathcal{O}(4^k m)$ for projections, resulting in $\mathcal{O}(4^k nm)$ time in total.

□

References

[1] Bryant, R. E. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers 35*, 8 (1986), 677–691.

[2] Choi, A., Kisa, D., and Darwiche, A. Compiling probabilistic graphical models using sentential decision diagrams. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013. Proceedings* (2013), L. C. van der Gaag, Ed., vol. 7958 of *Lecture Notes in Computer Science*, Springer, pp. 121–132.

[3] Dudek, J. M., Phan, V., and Vardi, M. Y. ADDMC: weighted model counting with algebraic decision diagrams. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020* (2020), AAAI Press, pp. 1468–1476.

[4] Dudek, J. M., Phan, V. H. N., and Vardi, M. Y. DPMC: weighted model counting by dynamic programming on project-join trees. In *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings* (2020), H. Simonis, Ed., vol. 12333 of *Lecture Notes in Computer Science*, Springer, pp. 211–230.

[5] Kwisthout, J., Bodlaender, H. L., and van der Gaag, L. C. The necessity of bounded treewidth for efficient inference in Bayesian networks. In *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings* (2010), H. Coelho, R. Studer, and M. J. Wooldridge, Eds., vol. 215 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 237–242.

[6] Robertson, N., and Seymour, P. D. Graph minors. III. planar tree-width. *J. Comb. Theory, Ser. B 36*, 1 (1984), 49–64.

[7] Somenzi, F. CUDD: CU decision diagram package release 3.0.0. *University of Colorado at Boulder* (2015).